A Modified Artificial Bee Colony Algorithm for Scheduling Optimization of Multi-aisle AS/RS System

Xiaohui Yan[1,2(✉)], Felix T. S. Chan[2], Zhicong Zhang[1], Cixing Lv[1], and Shuai Li[1]

[1] School of Mechanical Engineering,
Dongguan University of Technology, Dongguan 523808, China
Yxhsunshine@gmail.com

[2] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University,
Hong Hum, Kowloon, Hong Kong

Abstract. A modified artificial bee colony algorithm is proposed for solving the scheduling optimization problem of multi-aisle automatic storage/retrieval system. The optimization model of the problem is analyzed and founded, in which the sequence constraint of tasks and calculation of the number of aisles are more realistic. According to the features of the problem, the encoding and decoding strategies for solutions to MABC algorithm are redesigned. Probability selection-based updating method is also introduced to enhance the neighborhood search and preserve the good fragments. The experimental results show that MABC can obtain better results than PSO and GA algorithm, and is a competitive approach for AS/RS scheduling optimization.

Keywords: Automatic storage retrieval system · Modified artificial bee colony · Scheduling optimization

1 Introduction

Automatic Storage/Retrieval System (AS/RS) is an indispensable part of modern logistics. It has been widely used in manufacturing and logistics enterprises due to its advantages of high efficiency, economic space occupation and labor saving. In an AS/RS warehouse, S/R (storage/retrieval) machine is used instead of manual picking, which the saving labor cost usually accounts for 60–70% of the total warehouse operation cost [1]. Therefore, the scheduling optimization of S/R machine is the key point to AS/RS warehouse optimization. Many scholars have studied on it these years. Shiau et al. pointed out that picking scheduling optimization was a special case of Traveling Salesman Problem (TSP), and proposed a three-stage heuristic algorithm to optimize the order of the products to be picked [2]. Lerher et al. established travel time models for aisle transfer systems and shuttle-based systems for AS/RS warehouse [3, 4]. Ma et al. founded a multi-objective automated warehouse scheduling model and proposed an ensemble multi-objective biogeography-based optimization algorithm to solve it [5]. Cinar et al.

Y. Tan et al. (Eds.): ICSI 2020, LNCS 12145, pp. 94–103, 2020.
https://doi.org/10.1007/978-3-030-53956-6_9

investigated the scheduling of truck load operations in AS/RS, and proposed a priority based genetic algorithm to sequence the retrieving pallets [6].

However, there are also some shortcomings in existing studies. For example, in the multi-aisle AS/RS warehouse, one aisle corresponds to two storage racks on its both sides. Even if two products are on two different storage racks, the S/R machine can complete these two tasks without switching aisles if they are adjacent and located on two sides of the same aisle separately. Besides, S/R machines usually have more than one carrier and can hold several products at one time. However, it needs unoccupied carrier to pick up outbound products, which has certain requirements on the order of inbound and outbound tasks.

In this paper, the scheduling optimization model of multi-aisles AS/RS with multi-carrier S/R machine is established, and a modified artificial bee colony (MABC) algorithm is proposed to solve it. The new algorithm keeps the advantage of neighborhood searching of ABC algorithm and is also redesigned to adapt the AS/RS scheduling problem. Experimental results show that the proposed MABC algorithm has better performance than the particle swarm optimization (PSO) and genetic algorithm (GA) in solving this problem.

The rest of the paper is organized as follows. In Sect. 2, the multi-aisle AS/RS scheduling problem is introduced and its model is established. In Sect. 3, the MABC algorithm is proposed and described in detail. The experiment and discussion are given in Sect. 4 and conclusions are drawn in Sect. 5.

2 Scheduling Model of Multi-aisle AS/RS System

In this paper, we consider a multi-aisle AS/RS warehouse system which has been used in many enterprises. In this kind of warehouse, there are several rows of storage racks [7]. Usually, the two edges of the warehouse are one side rack, close to the wall, while the middle storage racks are two side racks back-to-back. There are passable aisles between the storage racks for S/R machines to travel, store and retrieve products. And the S/R machines can enter and exit freely at both ends of the storage racks. Each rack has a number of layers and columns, each of them has its own coordinates, corresponding to a storage unit which used to storage products. We use triples $[X, Y, Z]$ to represent the location of a storage unit, and Z, X, Y represent the serial number of the rack and the serial numbers of column and layer of the location in the rack respectively.

There is an I/O location for inbound and outbound operation. It is located at one side of the warehouse and denoted by the triple [0, 0, 1]. Most of the existing S/R machines have multiple carriers (such as forks) in order to improve efficiency, which can load more than one product at a time. We assume that the number of carriers is N. On one travel, the S/R machine picks up a maximum of N inbound products from the I/O location and places them in the designated location, then it picks up a maximum of N outbound products from the designated location and returns to the I/O point. In this paper, we mainly concern the total operation time for S/R machine to complete all tasks.

Generally, in order to maximize the efficiency of S/R machine, reduce the operation time, it is better to carry as many as possible products at the same time. In each route, the S/R machine should carry N storage products, complete N inbound tasks, and take out

N outbound products, complete N outbound tasks if the number of rest task is no less than N. Suppose there are m inbound tasks, n outbound tasks, $r = \max(\text{ceil}(m/N), \text{ceil}(n/N))$, where ceil($x$) stands for rounding the elements of x to the nearest integers towards infinity. It needs r routes to complete all inbound and outbound tasks. For situation that $m \neq n$, we can add $|m - n|$ virtual inbound or outbound tasks to make them be equal, which is more convenient for coding and calculation. The locations of virtual tasks are set as [0, 0, 0]. These tasks will be ignored when calculating the operation time.

As it has mentioned above, the AS/RS scheduling optimization problem can be regarded as a special case of TSP problem, we can construct its 0–1 integer programming model.

Define $e_{jk} = 1$, if the S/R machine travels from storage unit j (corresponding to task j) to storage unit k, otherwise, $e_{jk} = 0$. Define $a_{ji} = 1$, if the task j is executed in route i, otherwise, $a_{ji} = 0$. Define $b_{gi} = 1, c_{gi} = 0$, if the gth task in route i is an inbound task, else if it is an outbound task, $b_{gi} = 0, c_{gi} = 1$.

The goal is to minimize the total operation time.

$$min\, f = \sum_{i=1}^{r} \left(\sum_{j=1}^{m+n} \sum_{k=1}^{m+n} t_{jk} e_{jk} a_{ji} a_{ki} \right). \tag{1}$$

Among that, t_{jk} is the time for S/R machines moves from storage unit j to storage unit k.

The constraints are listed as below.

$$\sum_{i=1}^{r} a_{ji} = 1, \ \forall\, j \in 1, 2, 3 \ldots m + n, \tag{2}$$

$$\sum_{j=1}^{m} a_{ji} \leq N, \ \forall\, i \in 1, 2, 3 \ldots r, \tag{3}$$

$$\sum_{j=m+1}^{m+n} a_{ji} \leq N, \ \forall\, i \in 1, 2, 3 \ldots r, \tag{4}$$

$$\sum_{i=1}^{r} \sum_{j=1}^{m+n} a_{ji} = m + n, \tag{5}$$

$$\sum_{g=1}^{h} b_{gi} - \sum_{g=1}^{h} c_{gi} \geq 0, \ \forall\, i \in 1, 2, 3 \ldots r, \ \forall\, h \in 1, 2, 3 \ldots 2N. \tag{6}$$

Formulation (2) indicates that each task is executed only once. Formulations (3) and (4) are the load capacity constraint. It indicates that there can be no more than N inbound tasks and N outbound tasks are executed in each route. Formulation (5) grantees that all tasks are executed. Formulation (6) grantees that the S/R machine can pick up outbound products only when it has unoccupied carrier.

Suppose the storage racks are numbered start from 1, as it has mentioned in Sect. 1, the first rack is close to the wall. The first and second racks are separated by an aisle. The second and third racks are back to back close to each other. The third and fourth racks are separated by an aisle, and so on.

The time for S/R machine moves from storage unit j to storage unit k is calculated as follow.

$$t_{jk} = \begin{cases} max\left(W \times \left|X_j - X_k\right|/V_x, H \times \left|Y_j - Y_k\right|/V_y\right), & \text{if it doesn't need to swith aisles} \\ max((W \times min(\left|X_j - X_k\right|, 2C - X_j - X_k) + L \times \theta\left(Z_j, Z_k\right))/V_x, H \times \left|Y_j - Y_k\right|/V_y), & \text{else} \end{cases}, \tag{7}$$

where W is the width for each storage unit, H is the height of each storage unit, L is the width of each aisle. $\theta\left(Z_j, Z_k\right)$ is a function to calculate the number of aisles between the location of the *jth* and *kth* tasks. C is the number of columns for each storage rack. If the locations of task j and k are at the same storage rack or at the adjacent storage but are distributed on two sides of the same roadway (this condition can be expressed as $Z_j = Z_k \ or \ \left(\left|Z_j - Z_k\right| = 1 \ \&\& \ mod\left(min(Z_i, Z_i), 2\right) \neq 0\right)$), the S/R machine can move from location j to k without switching aisles. Otherwise, the S/R machine needs to switch aisles. Since the S/R machine can move from both ends of the aisles, it is necessary to consider from which end the distance is shorter. In addition, it is also necessary to consider the time required on switching aisles.

$\theta\left(Z_j, Z_k\right)$ is calculated as follows.

$$\theta\left(Z_j, Z_k\right) = \begin{cases} \left|Z_j - Z_k\right|/2 & if \ mod\left(\left|Z_j - Z_k\right|, 2\right) = 0 \\ \left|Z_j - Z_k\right|/2 - 0.5, & if \ mod\left(\left|Z_j - Z_k\right|, 2\right) \neq 0 \ \&\& \ mod\left(min\left(Z_j, Z_k\right), 2\right) \neq 0 \\ \left|Z_j - Z_k\right|/2 + 0.5, & if \ mod\left(\left|Z_j - Z_k\right|, 2\right) \neq 0 \ \&\& \ mod\left(min\left(Z_j, Z_k\right), 2\right) = 0 \end{cases} \tag{8}$$

3 Modified Artificial Bee Colony Algorithm

3.1 Artificial Bee Colony Algorithm

Artificial bee colony algorithm is proposed by Karaboga in 2005, which inspired by the behaviors of the bee colony searching food sources [8]. In ABC algorithm, it regards the searching space as the natural environment, and each solution of the problem represents a food sources to be exploited. The amount of nectar of the food sources corresponds to the fitness of the solution. There are three kinds of bees in ABC algorithm, employed bees, onlooker bees and scout bees. In ABC algorithm, half of the colony is employed bees the other half is onlooker bees. The number of scout bees is set to 1.

The process of ABC algorithm is also divided into the stage of employed bees, the stage of onlooker bees and the stage of the scout bee.

At the initialization stage, a set of food sources is randomly generated in the search space. The number of food sources equals half of the number of bees. The dimension of the food sources is the same with the problem to be solving. For each food source, there is a counter used to record the cumulative iterations for which it has not been improved.

At the employed bees' stage, each employed bee looks for a new food source near the original one according to formula (9).

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{k,j}\right). \tag{9}$$

Among that, x_i is the original food source, x_k is a randomly selected neighbor, v_i is the candidate food source newly produced, j is a randomly selected dimension. ϕ is a random number according to uniform distribution between [−1, 1]. If the new food source is better than the original one, then the employed bee turns to the new one and the original one is abandoned. Otherwise, the original one is kept and the corresponding counter is increased by 1.

At the onlooker bees' stage, the onlooker bee chooses a food sources to exploit depending on a probability related to the fitness of the food sources, seen as formula (10). The food source with higher fitness has larger probability to be chosen and may be chosen more than once.

$$P_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}. \tag{10}$$

After the food source is chosen, the onlooker bee will exploit a new food source nearby, just like it does at the employed bees' stage. Greedy selection and un-improved counter are also used.

At the scout bee's stage, if a bee's un-improved counter is larger than a predetermined parameter "*limit*", it indicates that the food source has been exploited out, the bee becomes scout bee, and will find a random food source in the searching area. At the same time, the counter is reset to 0.

Compared with PSO and other algorithm, ABC algorithm pays more attention to neighborhood search and obtains good results in many numerical and engineering optimization problems [9]. The optimal solution of scheduling optimization is usually obtained by neighborhood transformation of the suboptimal solution. Therefore, ABC algorithm may have better optimization potential in the AS/RS scheduling problem. However, we also need to redesign the encoding, decoding and updating methods of solutions to make the algorithm suitable for discrete AS/RS scheduling optimization problem.

3.2 Modified Artificial Bee Colony Algorithm

3.2.1 Encoding and Decoding Strategies

As mentioned above, the solution to the multi-aisle AS/RS warehouse scheduling problem is a sequence of inbound and outbound tasks. For the convenience of programming and solving, we use real number coding in MABC algorithm. The solution of the problem is a random sequence of the task numbers without repetition. However, according to the above constraints, the feasible solution also has certain requirements. The S/R machine must have an empty carrier to carry outbound tasks. As a result, in each route, the number of inbound tasks (including virtual task) must be larger than or equal to the number of outbound tasks from the first task to the end of the route. A repair mechanism is designed to ensure the feasibility of the solution.

a) Divide the solution into inbound task sequence and outbound task sequence according to the serial number.

b) Each time select N inbound tasks and N outbound tasks in sequence to form a route (including virtual tasks, and the number of tasks may be less than N at the last time due to the number of tasks may be not a multiple of $2N$ exactly). And then remove them from the task sequences.
c) In each route, the first task must be the inbound task. From the second one, compare the order of the first remaining inbound and first remaining outbound tasks in the original solution. If it is not against the rule following the order, then add the first task in order to the route. Otherwise, add the inbound task to the route.

Through this mechanism, all randomly generated solutions can be converted to the corresponding feasible solutions. This repair mechanism is not only used in MABC algorithm, but also used in the other comparison algorithms. Each time a new solution is produced, it needs to use this repair mechanism to convert it to a feasible solution before calculating its fitness.

3.2.2 Probability Selection-Based Solution Updating Strategy

The solution updating strategy of the original ABC algorithm is designed for continuous optimization problem. It learns from a random neighbor on a randomly selected dimension. The AS/RS scheduling optimization problem is a discrete optimization problem. We must redesign the solution updating strategy so that it can adapt to and solve the problem better. The neighborhood searching of ABC is modified as follows.

a) Select a neighbor solution randomly.
b) The values on dimensions which their values are the same with the neighbor solution are inherited into the same location of the new solution.
c) For the dimensions which their values are not the same, they are selected with a probability of 0.5.
d) For the selected dimensions, the values of the original food source on these dimensions are directly inherited into the same location of the new solution.
e) For the dimensions which are not selected, find the positions of their values in the neighbor solution, and insert them into the vacant positions of the new solution in order.

This solution updating strategy can guarantee that the solution newly produced is a non-repetitive task sequence, which can be transformed to a feasible solution of the AS/RS scheduling problem. On another hand, the solution is produced using neighborhood searching on the basis of the original solution. It keeps the advantage of original ABC algorithm. Both the neighborhood searching in employed bees and onlooker bees' phases use this strategy. In the scout bees' phase, a random solution is generated to keep diversity.

4 Experiments and Results

In this section, we tested the optimization ability of MABC algorithm on AS/RS scheduling problem and compared it with GA and PSO algorithm.

4.1 Test Instances

The width of each storage unit $W = 0.5$ m, The height of each storage unit $H = 0.8$ m, the width of each aisle $L = 3.0$ m, The numbers of the storage racks is 14. The numbers of columns and layers of each rack are 15 and 6. Accordingly, location X, Y and Z for the inbound and outbound tasks are distributed randomly from [1, 15], [1, 6] and [1, 14] separately. The movement velocity on the horizontal direction is 1.2 m/s. The movement velocity on the vertical direction is 0.4 m/s, which is close to the actual value. In this experiment, we generated three instances. In the first instance, the total number of tasks is 30, the number of inbound tasks m equals 14, number of outbound tasks n equals 16. In the second instance, m equals 22, n equals 18. In the third instance, m equals 26, n equals 24. The numbers of carriers of the S/R machine N of all instances are 2.

4.2 Parameters Setting

In this experiment, the population sizes of all algorithms are 50. Maximum number of function evaluations (FEs) is used as the terminated criterion [10, 11], and its value is 10000. In MABC algorithm, $limit = 50$. In PSO, learning factor $C1 = C2 = 2$, inertia weight ω decreases linearly from 0.9 to 0.4 [12]. The continuous version is used. The boundaries of all dimensions are [0, 1], and the maximum and velocity is 0.1 and minimum velocity is -0.1, the solutions are converted to desecrate solutions of AS/RS problem by rank of order (ROV) after each updating. In GA algorithm, crossover probability $p_c = 0.95$, mutation probability $p_m = 0.1$ [13]. In the crossover stage, some tasks may appear twice, and some others may be missing in the offspring solutions. These solutions will be checked and repaired to the sequence of tasks numbers without repetition. Greedy selection is also used in this stage. In the mutation stage, swap operator is used. The values of two randomly selected dimensions will be swapped with each other.

For all the three algorithms, the repair mechanism mentioned in Sect. 3.2.1 is used to make the solutions feasible and won't break the requirements of orders of inbound and outbound. All solutions will use this mechanism before being evaluated.

4.3 Results and Analysis

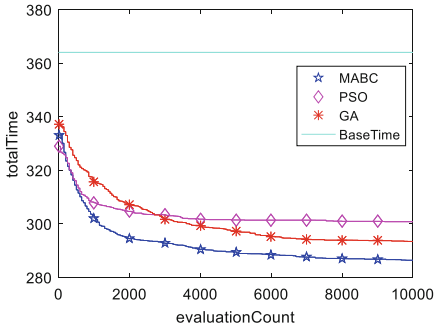
The results obtained by MABC, PSO and GA are listed in Table 1. It is intuitive that we can obtain an acceptable feasible solution if we sort all the tasks according to the serial numbers of racks (Z), and then repaired it according to the above repair mechanism. Therefore, we define this solution as the base solution and the time it takes as the base time. In the initialization phase of the three algorithms, all the solutions in the initial populations are generated by applying a swap operator on the base solution.

Each algorithm will run for 20 times independently on the instances. The best mean and standard are marked as bold. The mean convergence plots and boxplots of the final results are also given in Fig. 1.

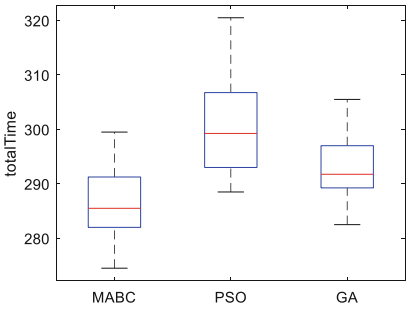
It is clear that MABC obtained the best results on all the three instances. Its convergence speed and accuracy are all the best from Fig. 1. The mean results obtained by MABC are reduced more than 20% compared with base time. PSO algorithm converges fast at the beginning but hardly improves after $FEs = 5000$, and obtained the worst results

Table 1. Results obtained by MABC, PSO and GA on the three instances (unit: second)

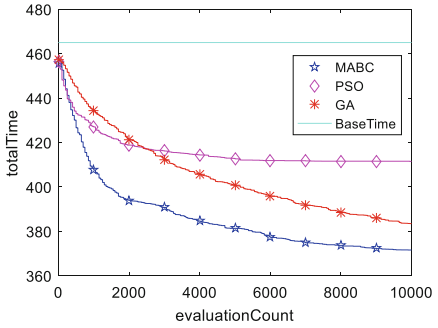
Instances	Base time	MABC		PSO		GA	
		Mean	Std	Mean	Std	Mean	Std
1	371.00	**286.3750**	6.7432	300.7750	9.0619	293.4000	**6.0188**
2	465.00	**371.5750**	**6.9609**	411.4750	10.4874	383.4750	10.3942
3	652.00	**508.7000**	**4.4290**	555.1750	10.2396	524.9750	8.0761



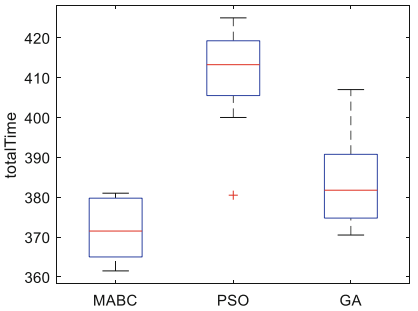
(a) Convergence plot on instance 1



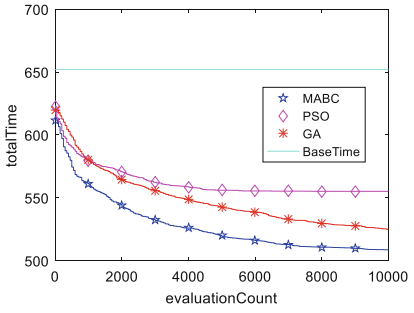
(b) Boxplot on instance 1



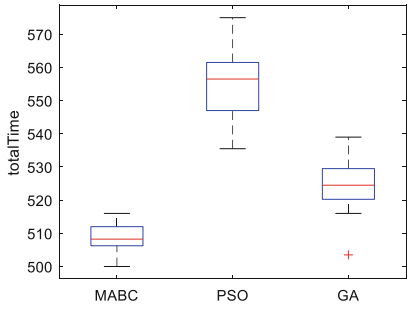
(c) Convergence plot on instance 2



(d) Boxplot on instance 2



(e) Convergence plot on instance 3



(f) Boxplot on instance 3

Fig. 1. Convergence plots and boxplots obtained by MABC, PSO and GA on the instances

among the three algorithms. The standard deviations of MABC are also the smallest on instance 2 and instance 3. On instance1, its standard deviation is only a little worse than GA, which can also be seen from the boxplots in Fig. 1. The results show that MABC is superior to PSO and GA and is suitable for solving AS/RS scheduling problem.

The locations of inbound and outbound tasks in instance 1are listed in Table 2. The best solution obtained by ABC algorithm on this instance is (I_5-O_{11}-I_9-O_{16}), (O_3-O_{12}), (I_{11}-I_7-O_{15}-O_6), (I_4-O_{14}-I_2-O_4), (I_{10}-I_3-O_{13}-O_7), (I_{12}-I_1-O_8-O_9), (I_8-I_{14}-O_{10}-O_1), (I_{13}-I_6-O_2-O_5), each bracket represents a route. And there are two virtual inbound tasks in the second route when calculating. The total operation time of this solution is 274.5 s.

Table 2. The locations of inbound and outbound tasks in instance 1

I_1(11, 3, 9)	I_2(12, 2, 5)	I_3(9, 6, 10)	I_4(3, 2, 4)	I_5(15, 4, 1)	I_6(4, 4, 14)	I_7(1, 3, 4)
I_8(15, 2, 9)	I_9(10, 5, 2)	I_{10}(5, 2, 8)	I_{11}(1, 2, 3)	I_{12}(7, 2, 10)	I_{13}(3, 3, 11)	I_{14}(14, 2, 11)
O_1(3, 1, 14)	O_2(7, 3, 13)	O_3(2, 1, 3)	O_4(10, 1, 6)	O_5(1, 1, 12)	O_6(8, 6, 4)	O_7(15, 6, 9)
O_8(9, 4, 10)	O_9(13, 4, 6)	O_{10}(11, 2, 13)	O_{11}(15, 3, 1)	O_{12}(2, 2, 2)	O_{13}(14, 6, 9)	O_{14}(9, 1, 3)
O_{15}(4, 5, 3)	O_{16}(5, 3, 1)					

5 Conclusions

The scheduling optimization of multi carrier S/R machine in multi-aisle AS/RS warehouse is introduced in this paper. A 0–1 integer programming model is founded, which considers the realistic constraint of orders to inbound task and outbound task. The calculation of the number of aisles between two positions of the adjacent tasks is also redefined due to the realistic placement of the storage racks. A modified artificial bee colony algorithm is proposed for solving this optimization problem. In MABC, the encoding and decoding strategy are redesigned, a probability selection-based updating strategy is also introduced. The modifications make the algorithm adapt to the features of the problem, while keeping the advantages of ABC algorithm's neighborhood searching. Three instances with 30, 40 and 50 tasks were employed to test the optimization capability of the algorithm. The results show that the MABC outperforms PSO and GA algorithm both on convergence speed and accuracy, and is a suitable approach for solving the AS/RS scheduling problem.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 61703102, 71971143, 71801045, 71801046), the National Key Research and Development Program of China (2018YFB1004004). The authors would like to thank The Hong Kong Polytechnic University Research Committee for financial and technical support.

References

1. Chen, T.L., Cheng, C.Y., Chen, Y.Y., et al.: An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. Int. J. Prod. Econ. **159**, 158–167 (2015)
2. Shiau, J.Y., Lee, C.M.: A warehouse management system with sequential picking for multi-container deliveries. Comput. Ind. Eng. **58**(3), 382–392 (2010)
3. Lerher, T., Potrc, I., Sraml, M., Tollazzi, T.: Travel time models for automated warehouses with aisle transferring storage and retrieval machine. Eur. J. Oper. Res. **205**(3), 571–583 (2010)
4. Lerher, T., Ekren, B.Y., Dukic, G., Rosi, B.: Travel time model for shuttle-based storage and retrieval systems. Int. J. Adv. Manuf. Technol. **40**(3), 101–121 (2015)
5. Ma, H., Su, S., Simon, D., Fei, M.: Ensemble multi-objective biogeography-based optimization with application to automated warehouse scheduling. Eng. Appl. Artif. Intell. **44**, 79–90 (2015)
6. Cinar, D., Oliveira, J.A., Topcu, Y.I., Pardalos, P.M.: Scheduling the truckload operations in automated warehouses with alternative aisles for pallets. Appl. Soft Comput. **52**, 566–574 (2017)
7. Dornberger, R., Hanne, T., Ryter, R., Stauffer, M.: Optimization of the picking sequence of an automated storage and retrieval system (AS/RS). In: 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, pp. 2817–2824 (2014)
8. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
9. Li, J.Q., Pan, Q.K., Duan, P.Y.: An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping. IEEE Trans. Cybern. **46**(6), 1311–1324 (2017)
10. Ma, L., Wang, X., Huang, M., Lin, Z., Tian, L., Chen, H.: Two-level master-slave RFID networks planning via hybrid multiobjective artificial bee colony optimizer. IEEE Trans. Syst. Man Cybern. Syst. **49**(5), 861–880 (2019)
11. Liang, J., Xu, W., Yue, C., et al.: Multimodal multiobjective optimization with differential evolution. Swarm Evol. Comput. **44**, 1028–1059 (2019)
12. Bonyadi, M.R., Michalewicz, Z.: Particle swarm optimization for single objective continuous space problems: a review. Evol. Comput. **25**(1), 1–54 (2017)
13. Kerh, T., Su, Y.H., Mosallam, A.: Incorporating global search capability of a genetic algorithm into neural computing to model seismic records and soil test data. Neural Comput. Appl. **28**(3), 437–448 (2017)